CLAIMS

For the convenience of the Examiner, all pending claims of the present Application are shown below.

1. A method for modeling a legacy computer system comprising:

identifying incidents of applications of the legacy computer system that output data; and

defining a control flow graph of the output incidents.

2. The method of Claim 1 further comprising:

identifying the value or type of the data fields associated with each output incident; and

attaching the value or type to the control flow graph.

3. The method of Claim 2 wherein identifying the value or type further comprises:

identifying output incidents of invariant data fields; and

attaching the value of each invariant data field to its associated control flow graph incident.

4. The method of Claim 2 wherein identifying the value or type further comprises:

identifying output incidents of variant data fields; and

attaching the type of each variant data field to its associated control flow graph incident.

5. The method of Claim 1 wherein the control flow graph comprises:

plural nodes having associated arcs, each node associated with an output incident.

6. The method of Claim 5 wherein a complete control flow graph of the application (N,A) is used to compute a directed graph (N_R, A_R) wherein:

n comprises a node in N_R if n, an element of N, starts an output process, stops an output process or outputs data; and

$<n_1, n_m>$ comprises an arc in A_R if n_1 and n_m are in N_R and a sequence of arcs $<n_1, n_2>$, $<n_2, n_3>$,. . .,$<n_{m-1}, n_m>$ is in A such that for i from 2 to m-1, n_i is not in N_R.

7. The method of Claim 6 further comprising:

defining the control flow graph as a formal grammar that describes the flow paths from each start command to the associated stop commands.

8. The method of Claim 1 further comprising:

associating the incidents with an Extensible Markup Language schema; and

creating a specification to modify the legacy computer system applications to provide output in Extensible Markup Language format.

9. The method of Claim 8 further comprising:

automatically modifying the legacy computer system applications in accordance with the specification.

10. A system for modeling an output application of a legacy computer system comprising:

a modeling engine interfaced with the legacy computer system, the modeling engine operable to analyze an application loaded on the legacy computer system to identify incidents within the application that output data from the legacy computer system; and

a control flow graph of the output operations within the applications.

11. The system of Claim 10 wherein the control flow graph comprises plural nodes, each node associated with an output incident.

12. The system of Claim 11 wherein a complete control flow graph of the application (N,A) is used to compute a directed graph (N_R, A_R) wherein:

n comprises a node in N_R if n, an element of N, starts an output process, stops an output process or outputs data; and

$<n_1, n_m>$ comprises an arc in A_R if n_1 and n_m are in N_R and a sequence of arcs $<n_1, n_2>$, $<n_2, n_3>$,. . .,$<n_{m-1}, n_m>$ is in A such that for i from 2 to m-1, n_i is not in N_R.

13. The system of claim 10 wherein the control flow graph of the output operations comprises as a formal grammar that describes the flow paths from each start command to the associated stop commands.

14. The system of Claim 10 further comprising a graphical user interface in communication with the modeling engine, the graphical user interface operable to display the control flow graph formal grammar and the incidents.

15. The system of Claim 14 wherein the graphical user interface further communicates with a mapping engine and an Extensible Markup Language schema, the mapping engine operable to map the incidents of the applications with the control flow graph formal grammar and the Extensible Markup Language schema.

16. **(Allowed)** A method for modeling a legacy computer system comprising:

defining a control flow graph of output incidents of applications of a legacy computer system;

wherein the control flow graph comprises plural nodes having associated arcs, each node associated with an output incident; and

wherein a complete control flow graph of the application (N,A) is used to compute a directed graph (N_R, A_R) wherein:

n comprises a node in N_R if n, an element of N, starts an output process, stops an output process or outputs data; and

$<n_1, n_m>$ comprises an arc in A_R if n_1 and n_m are in N_R and a sequence of arcs $<n_1, n_2>$, $<n_2, n_3>$,. . .,$<n_{m-1}, n_m>$ is in A such that for i from 2 to m-1, n_i is not in N_R.